SOFTWARE PROJECT PLAN

FOR

A Component Based Force & Moment Module (CLIC)

Task:	**AIC Internal Research: Automated Methods for Computational Fluid Dynamics**
Responsible Manager (RM):	Michael Aftosmis
Software Project Manager (SPM):	Michel Delanaye

I. Introduction

I.1. Background

In analyzing the aerodynamics of a configuration, it is generally necessary to back-out force and moment information both to gain insight into convergence behavior and to feedback into the "Loads and Lines" decision making process. Force and moment information may be required for an entire configuration, or may be confined to a subset of the components on a given configuration. While conceptually straightforward, post-processing this information on a case-by-case basis typically requires substantial expertise and can be extremely time-consuming.

Examples:

• Global forces

• Forces for group or selected components

• C_M of selected components with respect to an arbitrary moment center

• C_M of selected components with respect to an arbitrary line (defined by 2 points)

• C_p profile of selected components at body intersection with a plane (defined by 3 points)

The frequent requirement of using force and moment data for convergence assessment implies that a comprehensive module should be accessible as a client-side background executable as well as through a traditional intuitive front-end.

I.2. Scope

This module should take as input an annotated surface triangulation of a configuration, with triangles assigned to components on a triangle-by-triangle basis, and load information annotated to vertices in the triangulation. Output requests should be made with a file-based interface that can be modified either via a standard Unix editor or a "Forms" style interface. A primary design goal is that the module requires minimal memory and CPU so that it can be spawned remotely as a "thin client" from a remote server.

I.3. Input triangulation formats

Initially Cart3D style input triangulations should be implemented (binary and ascii), however, VRML and FAST format triangulations should be planned for.

I.4. Output formats

Output of force and moment data should be possible in either ascii or html. C_p should be output as *x, y* ordered data suitable for a scriptable 2-D plotter (like Xmgr). This architecture permits scripting to automatically produce hypertext compilations of results.

I.5. Documentation and Programming

The module needs to be fully documented in anticipation that it will be distributed on an unsupported basis. This includes both fully commented source code with version control and detailed electronic documentation. This is critical to ensure that future extensions of the package can be neatly and efficiently integrated. Third party extensibility should be considered a design goal. In anticipation of this release, the software and documentation must comply with ISO9001.AI.0003.1 for internally developed software to be released to external customers. The software and documentation should comply with the Level II specification.

I.6. Milestones

Dates and detailed milestones will need to be worked out, but they should include:

- 5.1 Overall Architecture and Interface Design
- 5.2 Data Structures
- 5.3 Rudimentary Implementation
- 5.4 Documentation
- 5.4 Full implementation
- 5.5 Final Q/A Checking (memory, portability, documentation, etc.)
- 5.6 Compliance with ISO9001.AI.0003, Complete ARC 310, NF 1676, AI-310-A, AI-02

II. Definitions

II.1 Load, moment, axis definitions

The coordinates of the vertices of the triangulation (the model) are by default assumed to be defined in the *body frame (d*enoted *Fb)* as presented in Fig. II.1. The aerodynamic frame (denoted *Fa)* is the frame attached to the flow (see Fig. II.1).

In the aerodynamic frame (Fa), we calculate the forces denoted by:

$$C_F^{\,a} = \begin{bmatrix} CX^a \\ CY^a \\ CZ^a \end{bmatrix} = \begin{bmatrix} -CD \\ Cy \\ -CL \end{bmatrix}$$

In the body frame (*Fb*), the aerodynamic coefficients are expressed as:

$$C_F^{\,b} = \begin{bmatrix} CX^b \\ CY^b \\ CZ^b \end{bmatrix} = \begin{bmatrix} -CA \\ CY \\ -CN \end{bmatrix}$$

The symbol *CA* is called coefficient of the axial force and *CN* the coefficient of the normal force.

The following relation hold:

$$\begin{bmatrix} -CA \\ CY \\ -CN \end{bmatrix} = \begin{bmatrix} -CD\cos\alpha\ \cos\beta - Cy\sin\beta\cos\alpha + CL\sin\alpha \\ -CD\sin\beta + Cy\cos\beta \\ -CD\cos\beta\sin\alpha - Cy\sin\alpha\sin\beta - CL\cos\alpha \end{bmatrix}$$

alpha and beta are the classical incidence and sweep angles as defined in Fig. II.1 between the aerodynamic and body frames.

III. Overall Architecture

III.1 Preliminary program interface

Clic can be run in standalone mode or can be started as a separate thread from any remote machine by issuing system calls from the running remote server code.

Startup command on Unix:

```
system(“rsh $CLIENTNAME $CLICDIR/clic -i inputFileDir
                                      -t triangulation_file
                                      -c control_file
                                      -d outputFileDir
                                      -html “)
```

The -i option specifies the directory where input files can be found.

The -t option specify the triangulation file name which can also be specified in the control file. The one provided on the command line however takes precedence on the file name provided in the control file.

The -c option is mandatory. An error message will be issued if the control file is not found or neither specified on the command line.

The -d option specifies the directory where out put files are written.

The -html option can be used to output the load and moment calculations in an html file

III.2 Input file formats

Data are passed by two files. The first file (*triangulation file*) should contain the annotated triangulation, pressure distribution (Cp), and component information, assumed written by fortran unformatted or fortran free format (ascii). The second file is a *control file* containing the different loads and moments requested for each components, group of components or for the entire geometry.

Within clic, a separate triangulation is created for each component based on the complete triangulation read from the triangulation file.

III.2.1 triangulation file format

The following triangulation file format will be assumed for the first version of the code:

The coordinates of each vertex as well as the Cp are provided. Then the list of three indices of each triangles is provided. Finally, the component number associated with each triangle is given.

```
  nVerts, nTri            <- integer, integer
   x_1, y_1, z_1 ,Cp1 <- Geometry of Vertex 1,Cp at vertex 1
   x_2, y_2, z_2 ,Cp2 <- Geometry of Vertex 2,Cp at vertex 2
         .
         .
         .
 x_nVerts, y_nVerts, z_nVerts, Cp_nVerts
 v1_t1, v2_t1, v3_t1  <- Vertices of triangle 1 (int, int, int)
 v1_t2, v2_t2, v3_t2  <- Vertices of triangle 2 (int, int, int)
 v1_t3, v2_t3, v3_t3  <- Vertices of triangle 3 (int, int, int)
         .
         .
         .
v1_nTri, v2_nTri, v3_nTri <- Vertices of last triangle (int, int, int)
1 1 1 1 1 . . . 1 1 1 2 2 2 . . . 2 2 2 <-component numbers of all
triangles in the configuration
```

III.2.2 Control file format

The control file provides all necessary information and requests to perform the loads and moments calculation. Each line of the file must start by a blank, a carriage return ‘\n’, a ‘#’ sign specifying a comment, or a key command. The information provided on the line after a key command is interpreted depending on the latter. Key commands can be:

```
Title
Triangulation_file_name
Triangulation_X_axis_is
Triangulation_Y_axis_is
Triangulation_Z_axis_is
Incidence_Angle
Sweep_Angle
Component_Name
Component_Group
Reference_Area
Reference_Length
Force_Info
Moment_Point_Info
Moment_Line_Info
```

```
   Cp_Distribution
```

The keyword “*all*” means that a command applies to all the components. The name “*all*” should therefore never used as a component name.

A ‘\’ can be used as a continuation mark.

The description of each of the keys and their usage are described in the following.

A title can be provided:

Example of file line:

```
Title Load Calculation for an aircraft configuration
```

The name of the file containing the triangulation can be provided. If such a file name is also provided on the command line, it will take precedence on the one specified in the control file.

Example of file line:

```
Triangulation_file_name myTriangulation.file
```

Axis of the geometry are by default assumed to be the *body frame axis* denoted *Fb.* The user can however defined its own body axis with respect to the default one by the following command

Example of file line:

```
Triangulation_X_axis_is -Xb # specify that the X axis of the
                                 # triangulation is actually oriented
                                 # in the opposite direction of the default
                                 # X body frame axis
Triangulation_Y_axis_is +Zb#  specify that the Y axis of the
                                 # triangulation is actually oriented
                                 # in the direction of the default
                                 # Z body frame axis

Triangulation_Z_axis_is  Yb # specify that the X axis of the
                                 # triangulation is actually oriented
                                 # in the direction of the default
                                 # Y body frame axis
```

Incidence and sweep angles must be provided (*unit is assumed to be degrees*)

Example of file line:

```
Incidence_Angle     10   # a incidence  angle is 10 degrees
Sweep_Angle         15   # b sweep  angle is 15 degrees
```

A name can be associated with a component, either this name or the component number can be used to denote the component

Example of file line:

```
 Component_Name  flap 9          # Component number 10 has been
                                 #  associated to name ''flap''
```

A name can also be defined to refer to a group of components, a list of components is provided as a list of component numbers or previously defined names, an error message will be issued if a name hasn't been previously defined.

Example of file line:

```
Component_Group  wing 10 11 22 flap # the component group wing contains
                                    # components 10, 11 and flap
```

A reference length and a reference area must be provided for each component or group of components.

Example of file line:

```
Reference_Area   A  wing 8  # this reference area is used for all
                           # components defined with the name wing
                           # and for the component denoted by number 8
Reference_length  L wing 8  # this reference area is used for all
                           # components defined with the name wing
                           # and for the component denoted by number 8
```

An error message will be issued if a reference length or area is not provided for a component.

Forces information can be provided for a single component, for a group of components and by default for the whole geometry.

Example of file line:

```
Force_Info    wing 8 10  flap  # Force information will be
                                    # separately output for the group
                                    # component denoted by wing, for the
                                    # single components number 8, 10 and for
                                    # the one denoted by flap.
```

Any number of Force_Info request commands can be issued

Moments with respect to a point can be requested in the same way for a single component, for a group of components and by defeault for the whole geometry.

Example of file line:

```
Moment_Point_Info  x y z  wing 8 10  flap
                                    # Moment with respect to a point of
                                    # coordinates x1 x2 x3 will be
                                    # separately output for the group
                                    # component denoted by wing, for the
                                    # single components number 8, 10 and for
                                    # the one denoted by flap.
```

Moments with respect to a line defined by two points can be requested in the same way for a single components, for a group of component and by default for the whole geometry.

Example of file line:

```
Moment_Line_Info  x1 y1 z1 x2 y2 z2 wing 8 10  flap
                                    # Moment with respect to a line defined
                                    # by two points of coordinates
                                    # (x1 y1 z1) and
                                     # (x2 y2 z2)
                                    # separately output for the group
                                    # component denoted by wing, for the
                                    # single components number 8, 10 and for
                                    # the one denoted by flap.
```

Cp distributions can also be requested for a cutting plane through a component group, or a single component. The cutting plane is defined by three points.

Example of file line:

```
Cp_Distribution  x1 y1 z1 x2 y2 z2 x3 y3 z3 wing 8 10  flap
```

III.3 Output file formats

One output file named `LoadMoment.info` contains the information regarding the load and moment calculations requested for each component or group of components. An html version of the file will be created `LoadMoment.html` when providing the option -html on the command line. A file is written with each Cp distribution requested in a plottable x,y ordered format; the name of the file will be: `cpDistribution_$.dat`, with $ being the component or group name, or number if no name was specified for this component or group.

III.4 Data structures

The include fle dataypes.h defines the data structures used in the code.

```
#ifndef DATATYPES_H
#define DATATYPES_H

#include "circlist.h"  /* circular list include file */

#define DIM 3
#define STRINGLEN  256 /* line of an input string */

typedef enum {X, Y, Z}     axis; /* axis annotation    */
typedef enum {FALSE, TRUE} bool; /* boolean definition */


typedef int    INT;          /* integer type definition */
typedef double REAL;         /* real type definition    */
typedef REAL   point[DIM];   /* point definied by DIM real coordinates*/

/* Flow Case Structure type definition */
typedef struct FlowCaseStructure tsFlowCase;
typedef tsFlowCase           * p_tsFlowCase;

/* Geometry component structure type definition */
typedef struct ComponentStructure tsCmpt;
typedef tsCmpt                * p_tsCmpt;


/* Triangulation of each component structure type definition */
typedef struct triangulationStructure tsTrig;
typedef tsTrig                    * p_tsTrig;

/* load and moment struture type definition */
typedef struct ForceMomentCpDistriStructure tsFMCp;
typedef tsFMCp                          * p_tsFMCp;

/* Forces data structure */
typedef struct forcesStructure  tsForces;
typedef tsForces            * p_tsForces;

/* Point Moment data structure */
typedef struct PointMomentStructure  tsPointMmt;
typedef tsPointMmt               * p_tsPointMmt;


/* Line Moment data structure */
```

```
typedef struct LineMomentStructure  tsLineMmt;
typedef tsLineMmt               * p_tsLineMmt;

/* Pressure coefficient distribution structure */
typedef struct CpDistributionStructure  tsCpDist;
typedef tsCpDist                    * p_tsCpDist;

struct forcesStructure{ /* Forces structure */
 REAL       CD,Cy,CL;  /* -> Forces in aerodynamic frame Fa           */
 REAL       CA,CY,CN;  /* -> Forces in body frame Fb                  */
};

struct PointMomentStructure{ /* Point moment structure */
 point      x;         /* -> reference point coordinates              */
 REAL       Cma[DIM];  /* -> moment in aerodynamic frame Fa           */
 REAL       Cmb[DIM];  /* -> moment in body frame Fb                  */
};

struct LineMomentStructure{ /* Line moment structure */
 point      x1, x2;    /* -> reference points coordinates                  */
 REAL       Cm;        /* -> moment with respect to the line defined by x1x2 */
};

struct CpDistributionStructure{ /* Cp distribution structure */
 point  x1, x2, x3;  /* -> reference points coordinates
                           (cutting plane)                 */
 INT       np;      /* -> np of points of the distribution */
 REAL    * s;        /* -> curvilinear abscissae of sample points
                           (first point s[0]=0)  */
 REAL    * Cp;       /* -> Cp at sample points    */
};

struct FlowCaseStructure{ /* Flow Case general structure */
 void    * info;        /* -> pointer to an arbitrary info object */
 char      title[STRINGLEN]; /* -> title                 */
 REAL      alpha, beta; /* -> incidence and sweep angles */
 INT       AxisTransfo[DIM]; /* -> axis transformation        */
 INT       nCmpts;      /* -> # of components of groups of components */
 tsCmpt  * Cmpts;       /* -> array of components        */
 int (*readTriangulation)(p_tsFlowCase, void *info);
 /* pointer to a function which reads the input file defines
     the triangulation and breaks it component by component and
     associates an index map for each component triangulation,
     all necessary information are provided by the info object */
};


struct triangulationStructure{ /* Component triangulation data structure   */
 void     * info;          /* -> pointer to an arbitrary info object */
 p_tsCmpt   p_Cmpt;        /* -> pointer to parent component */
 INT        nVerts, nTri;  /* -> # of vertices and # of triangles of
                                 the triangulation */
 point    * Verts;         /* -> coordinates of vertices */

 REAL     * Cp;            /* -> value of Cp at the vertices */
 INT      * Tri[3];        /* -> triangle definition     */
};

struct ForceMomentCpDistriStructure{ /* Force Moment CpDistri  data structure
*/
 void         * info;      /* -> pointer to an arbitrary info object      */
 p_tsCmpt       p_Cmpt;    /* -> pointer to parent component              */
 tsForces       forces;    /* -> forces in body and aerodynamic frame     */
 INT            nPointMmt; /* -> # point moment calculations              */
 INT            nLineMmt;  /* -> # point moment calculations              */
 INT            nCpDistri; /* -> # of Cp distribution                     */
 tsPointMmt   * pointMnt;  /* -> Point moments calculations               */
 tsLineMmt    * lineMnt;   /* -> Line  moments calculations               */
 tsCpDist     * CpDistri;  /* -> Cp Distribution                          */
};

struct ComponentStructure{ /*  component general structure */
 void          * info;             /* -> pointer to an arbitrary info
                                         object      */
 INT             no;               /* -> component number                  */
 char            name[STRINGLEN];  /* -> name of the component          */
 p_tsCmpt        p_Cmptgroup;      /* -> pointer to a parent cmpt group */
 INT              nCmptinGroup;     /* -> if the cmpt is a group,
                             # of components in the group component */
 p_tsCmpt      * GroupCmptList;    /* -> if the cmpt is a group,
                             define list of components otherwise NULL*/
 p_tsFlowCase     p_FlowCase;       /* -> pointer to parent flow case       */
 REAL            Lref, Aref;       /* -> reference length and area        */
 p_tsTrig        p_Trig;           /* -> pointer to a triangulation, NULL
                                         if a component group */
 tsFMCp          FMCp;             /* -> Force, Moment and Cp calculations */
```

```
};

#endif
```